LAB 7

PROJECT EXPLORATION MISSION: INTEGRATION AND IMPROVEMENT (MISSION 2)

INTRODUCTION

In this lab, you will conduct your second mission with the CyBot mobile robot, building on your first mission and integrating and improving code and functionality based on what you have been learning and implementing. The basic mission is the same: move the CyBot through a test field, detect and localize objects, and avoid objects as you identify and navigate to an object in the test field (in this case, the smallest width object). This mission continues to represent primary capabilities you will implement in your final lab project. Your work in this lab will include:

- Moving the CyBot as in Labs 2 and 3 in autonomous and/or manual mode
- Communicating between the CyBot and PC as in Labs 3-6
- Integrating your own code for the CyBot-PC UART into the mission application to replace pre-compiled library functions as in Labs 5 and 6
- Scanning for objects (still using the pre-compiled CyBot Scan library in this lab)
- Improving your understanding of operational aspects of the CyBot
- Improving your program and CyBot performance

In later labs, you will learn how to write your own code for scanning for objects that uses microcontroller peripherals to interface with the sonar (PING) and infrared (IR) sensors and servo. For the project, you will use your own code. Until then, the Scan library will let you continue experimenting with more capabilities as well as limitations of the CyBot in the test field. As an aside, think about this quote by composer Igor Stravinsky: "The more constraints one imposes, the more one frees one's self. And the arbitrariness of the constraint serves only to obtain precision of execution." In other words, the more constraints you have, the more creative you can be (e.g., https://jonathanvieker.com/increase-your-creativity-with-constraints-part-i/). This applies to designing an embedded system and writing code as much as music or other activities.

You will develop an embedded application that can: 1) identify the smallest width (tall) object in the test field, and then 2) navigate in autonomous mode to that object while avoiding obstacles, optionally using manual mode as needed.

REFERENCE FILES

There are no new reference files for this lab. Use previous files, quick reference sheets and datasheets as appropriate.

PRELAB

See the prelab assignment in Canvas and submit it prior to the start of lab.

STRUCTURED PAIRING

You are expected to continue to use structured pairing in this lab and in future labs. It was introduced in Lab 2.

BACKGROUND

Review the mission ground rules given in Lab 3.

The datasheets for the IR and PING sensors are provided on the Lab Manuals and Lab Resources pages in Canvas. These sensors will be examined in more detail in later labs.

IR Sensor

The infrared (IR) sensor is an electro-optical device that emits an infrared beam from an LED and has a position sensitive detector (PSD) that receives reflected IR light. A lens is positioned in front of the PSD in order to focus the reflection before it reaches the sensor. The PSD sensor is an array of IR light detectors, and the distance of an object can be determined through optical triangulation. The location of the focused reflection on the PSD is translated to a voltage that corresponds with the measured distance. The IR distance sensor is designed to measure distances from 9 – 80 cm. However, the IR sensor can only moderately accurately display a distance value for an object 9 – 50 cm away. Objects must be placed far enough apart to provide a measurable gap between them. This gap may not be measurable by the PING sensor, but the tighter beam of the IR distance sensor will generate data consistent with a physical gap.

The IR sensor is read using the analog to digital converter (ADC) module on the microcontroller. The analog voltage from the sensor is converted to a raw digital value by the ADC. The relationship between the voltage value and the measured distance is shown in the sensor's datasheet. This is the focus of Lab 8.

PING Sensor

The range of the PING SONAR sensor is 2 cm to 3 m. SONAR (Sound Navigation and Ranging) uses an ultrasonic burst (well above human hearing range) to determine the presence and distance of objects. SONAR is a term that is valid for both air and water. The frequencies of the pulses emitted are different for these two mediums. The sensor emits 40 KHz pulses and receives a sound wave reflected from an object. The distance is calculated by determining the time between emitting a sound pulse and receiving an echo. The echo is translated into distance given the speed of sound in a particular medium. In our case, the medium is air. While air temperature does affect the speed of sound, for our purposes we will assume the speed of sound to be constant at 340 m/s or 1130 ft/sec. The PING sensor will only report one echo for any given pulse. This one echo is the first received to meet the minimum threshold used to discriminate between noise and a proper echo. The pulse does spread as it travels away, so the first reflection may come from an object that is not directly in front of the sensor. Clutter affects the usefulness of the sensor. The composition and shape of objects affects the amount of sound that gets reflected back to the sensor. A soft material like fabric will absorb more sound than a hard material like steel. It is possible to angle a box so that sound waves will reflect away from the sensor, so the box may "disappear" like the B2 or F117 US Air Force airplanes.

The PING sensor is read using a special mode of the timer module on the microcontroller. The timer module measures the time taken to receive the echo, from which distance is calculated. This is the focus of Lab 9.

Here are strengths and weaknesses of each sensor.

Sensor Characteristics	Strengths	Weaknesses
PING Sensor	• Does not need to be calibrated • Measurements have relatively little noise	• "Cone" of detection is relatively wide • Compared to the IR sensor, its angular resolution is coarser • There may be reflections from the floor • If objects are too close together, they look like one large object
IR Sensor	• Narrow "cone" of detection • Has fine-grained angular resolution • Can distinguish between objects even if they are close together	• Needs recalibration if conditions change (e.g., different CyBot, different object material) • Compared to the PING sensor, measurements are noisier, due to various factors • Averaging multiple measurements at a given scan angle is a good way to reduce the noise

Object detection is done by distinguishing between distance measurements that pertain to objects in contrast to background noise. Distance measurements are needed to detect the presence of an object, and from these measurements, the edges of the object can be estimated at particular angular positions.

PART 1: IMPROVE OBJECT DETECTION AND LOCALIZATION

Review your solution to Lab 3 Part 2 (Collect Sensor Data and Identify Objects in the Data). In this part of the lab, you will improve your capability, as needed, to detect and localize an object in the test field.

The terminology related to object recognition may vary, and specific terms may have specific meanings in a particular context. For our purposes, object detection and localization are essentially the same task, which is to locate the presence of an object in a space. Distinct objects are identified in lab by finding the edges of an object.

The following approach is recommended to combine the strengths of the two sensor types for object detection and localization:

(1) Follow the servo calibration procedure or use previously calculated values for the calibration variables in your main program. Remember that servo calibration is specific to each bot.
(2) Make a scan from 0 – 180 (or 360) degrees. Take multiple measurements at each angle in the scan.
(3) At each position at which measurements are taken, average several raw IR values.
(4) Use the IR values to detect the edges of each object, i.e., the angles for the starting and ending edges.

(5) Make a second scan that points the PING sensor at the midpoint between the edges of an object (i.e., halfway between the angles of the starting edge and ending edge). Then measure the distance to that object using the PING sensor. Do this for each object detected in the initial scan.

Carefully review the functions and descriptions given in cyBot_Scan.h for the Scan library.

CHECKPOINT:

Display accurate object information in PuTTY.

PART 2: CALCULATE ACTUAL LINEAR WIDTH OF TALL OBJECTS

Review your Lab 3 functionality. In this part of the lab, you will update or improve your code to calculate actual object width.

Radial width (the difference between the angles of the starting and ending edges) is different than actual linear width, a property of the object. Geometry or trigonometry can be used to calculate or estimate the linear width. In this mission and the project, you will need to use linear width to accurately identify a narrow or wide object, because objects will be at varying distances from the CyBot in the test field.

CHECKPOINT:

Display accurate object widths in PuTTY.

PART 3: INTEGRATE YOUR UART FUNCTIONS

In previous labs, you developed your own code to initialize and use the UART device to send and receive information between the CyBot and PC. Your code replaced the pre-compiled UART library given to you earlier in the semester. Integrate the UART code you developed into the mission application. It's your choice whether you use the polling-based receive function or interrupt-driven receive. You may want to start with the polling-based non-blocking receive function, as this may simplify testing and debugging the integration of code in the mission application.

CHECKPOINT:

Object information is displayed in PuTTY using your own UART functions.

Part 4: Drive to the Smallest Width Object While Avoiding Other Objects

The test field will be populated with several tall objects. One of the tall objects will have a smaller width than the others. Some of the tall objects could be a composite object created by clustering a number of tall objects very close together. The test field will also include short objects that can only be detected by the CyBot bump sensors.

Using your improved object detection capability from Part 1, extend your embedded application so that the CyBot identifies and navigates to within 10 cm of the object with the smallest width, while navigating around objects it encounters. The CyBot should drive using an autonomous mode, i.e., it automatically scans and moves as needed to navigate to the smallest width object.

Checkpoint:

Object information is displayed in PuTTY, and the CyBot moves autonomously to within 10 cm of the object with the smallest width, while navigating around objects it encounters along the way.

2 Bonus Points:

In autonomous mode, the CyBot reaches the smallest width object within 2 minutes.

(Optional) Part 5: Drive Using Autonomous and/or Manual Mode (2 Bonus Points)

An embedded system may find itself in a situation where it doesn't know what to do. In such situations, it is useful to allow a human to take over control at least briefly to get back to a state that the system understands and can proceed autonomously.

Implement manual mode with the following or similar functionality:

> Let the user drive the CyBot using the 'w', 'a', 's', 'd' keys of the PC keyboard, and use the 'm' key to initiate a 180-degree scan. The user needs to be able to interpret the sensor data to manually drive within 10 cm of the smallest width object. Using a GUI to help interpret the sensor data is allowed, but not required. During manual driving mode, only sensor data can be used for navigating the CyBot. In other words, the user will not be allowed to look at the CyBot.

The user should be able to switch back and forth between autonomous and manual modes at any time during the mission. For example, the 't' key could be used to "toggle" between these modes.

Checkpoint:

The CyBot toggles between autonomous and manual modes when requested by a TA and reaches the smallest width object within 3 minutes. The user must drive during manual mode.

DEMONSTRATIONS:

1. **Functional demo of a lab milestone –** Specific milestone to demonstrate in Lab 7: Checkpoint for Part 4
2. **Debug demo using debugging tools to explain something about the internal workings of your system –** The TA will announce any specific debugging requirements at the start of lab; otherwise you will create your own debug demo based on your needs and interests in the lab.
3. **Q&A demo showing the ability to formulate and respond to questions –** This can be done in concert with the other demos.